Head Tracking Using Stereo

Daniel Russakoff, Martin Herman
Information Access Division
National Institute of Standards and Technology *
Gaithersburg, MD 20899

Abstract

Head tracking is an important primitive for smart environments and perceptual user interfaces where the poses and movements of body parts need to be determined. Most previous solutions to this problem are based on intensity images and, as a result, suffer from a host of problems including sensitivity to background clutter and lighting variations. Our approach avoids these pitfalls by using stereo depth data together with a simple human torso model to create a head tracking system that is both fast and robust. We use stereo data to derive a depth model of the background which is then employed to provide accurate foreground segmentation. We then use directed local edge detectors on the foreground to find occluding edges which are used as features to fit to a torso model. Once we have the model parameters, the location and orientation of the head can be easily estimated. A useful side effect from using stereo data is the ability to track head movement through a room in three dimensions. Experimental results on real image sequences are given.

1 Introduction

Human head tracking has been an area of active research in computer vision for several years. Head position and orientation are important parameters for a variety of applications, including telepresence [1], face recognition [2], voice recognition, and perceptual user interfaces [15]. Until recently, real-time stereo depth data has been unavailable on the commercial market. As a result, most previous approaches to head tracking deal exclusively with intensity images [3], [4], [5], [6], [7], [9]. These approaches are notoriously sensitive to environmental factors that affect intensity values such as changes in illumination, shadowing, or background clutter. Stereo depth calculations, however, do not encounter these problems. This observation, coupled with recent advances in stereo hardware which allow us to gather depth data in real time [10], suggests a new approach to the head tracking problem.

There has been very little work done on head tracking using stereo. Some systems ([11],[8]) use stereo but still rely heavily on skin-tone pixel extraction, an intensity-based measure subject to all of the aforementioned problems. One approach ([14]) uses stereo data only but their complicated models prevent tracking of rapid movements and require the user to move slowly. In addition, this algorithm also requires a manual initialization step.

Our approach is thus to use stereo data to perform a more accurate foreground segmentation. We then fit a simple torso model to the foreground, looking specifically for the occluding edges of the shoulders. Because the model is so simple, we can perform the fit to each frame separately without having to use traditional tracking techniques to limit the search space. This means that our tracker will not get confused as easily by rapid movements or temporary occlusions. Another important benefit from using stereo depth data is that, once we find the head in the depth image, if we know the cameras' focal lengths and baselines, we can determine its position in 3D coordinates.

The paper is organized as follows. Section 2 will discuss the algorithm for segmentation in detail. Section 3 will describe our torso model and its parameters, while Section 4 will discuss how we acquire that model in each frame. Section 5 presents our head localization algorithm. Our results and conclusions will be presented in Sections 6 and 7, respectively.

2 Segmentation

Our system begins with a segmentation of the human figure in the foreground of our image sequence. Conventional approaches using intensity images [7] create a Gaussian model of the intensity over a certain interval of time of each pixel in the background and then determine whether a pixel is part of the fore-

*Commercial equipment and materials are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

ground based on its distance from the background in the chosen color space. This method has two important limitations. First, it is extremely sensitive to variations in lighting conditions. For example, if the lighting suddenly changes, the background model is no longer valid and the resulting segmentation is incorrect. Similarly, the effects of shadows are very difficult to handle. If the foreground figure casts a shadow, the darkened region could differ enough from the background to be classified as foreground. In addition, if the foreground figure happens to be similar in color to the background, it will be classified as background.

2.1 Using Depth Data

The use of stereo eliminates the aforementioned problems. With depth images, we proceed as before, modeling each background pixel as a Gaussian with a mean μ and a standard deviation σ. This time, however, we build the model with depth instead of intensity values. Closely following the work of [12], once we build a depth model of the background, we can identify the foreground as any region where the depth is sufficiently closer to the camera than the background. This is much more physically intuitive than the intensity segmentation and more accurate as well. Because the nature of the stereo correlation calculation makes it insensitive to color, shadows, or lighting variations, we do not have to worry about the previous problems.

The segmentation, however, is not quite this simple. Stereo matching is extremely sensitive to image texture. In our case, the correlation-based stereo system has a great deal of difficulty operating in regions where there is little texture. For example, consider a blank wall. A stereo system attempting to correlate pixels in such an untextured region will have a difficult time finding the correct matches as all pixels look alike. The result is an area of incorrect matches yielding disparities more or less randomly distributed throughout a range dependent on the size of the correlation window. This noise, depicted in Figure 1 is neither Gaussian nor white, making it very difficult to model. Unfortunately, this adversely affects our segmentation as we cannot effectively model the background in regions without adequate texture. We can, however, identify those background pixels that are unreliable with a simple test of our model's standard deviation: if $\sigma_{ij} > \beta$ where β is a user-defined threshold (we used $\beta = 2$). To combat this problem in untextured regions, we've devised our own segmentation scheme, closely related to [12] but with an important additional validation step.

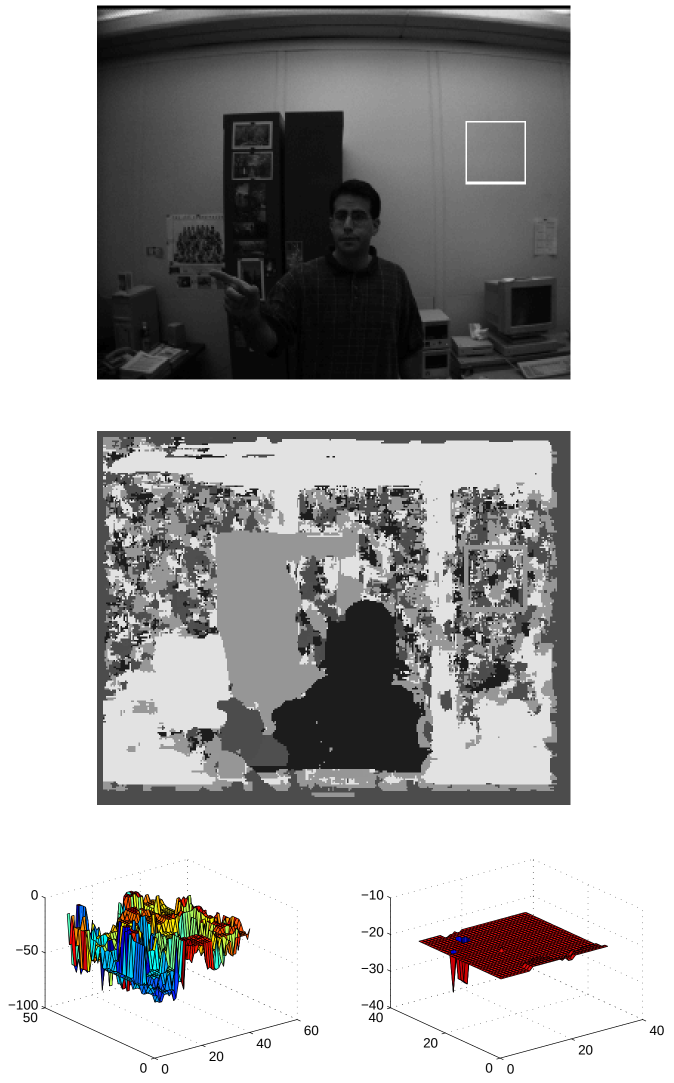


Figure 1: TOP: Sample input image from camera. White square highlights region with little texture. MIDDLE: Disparity image from the Digiclops^{TM} Stereo System from Point Grey Research, Inc. Note noise in the highlighted region. BOTTOM LEFT: Reconstruction of physical surface based on disparity values in highlighted region (negative z axis trends away from camera). BOTTOM RIGHT: Reconstruction of physical surface based on region centered inside of the human figure in the middle.

2.2 Surface Validation Segmentation

Once we've modeled the background, we face the problem of picking out the foreground in a new disparity[1] image DI. In the case where a pixel of the foreground DI_{ij} is in front of a reliable background pixel ($\sigma_{ij} < \ beta$), we've demonstrated that the segmentation is simple. All we must say is that a pixel is part of the foreground if the disparity value at that point is in front of the mean background by more than a standard deviation. When we are dealing with an unreliable background pixel ($\sigma_{ij} \geq \ beta$), things get much more complicated. In these cases, since μ_{ij} is not a reliable representation of depth, we cannot know based only on the value of DI_{ij} whether that pixel is in front of the background or not. Here we make an important assumption: the foreground figure must consist of a smooth blob of pixels with similar disparity values. In other words, it should be distinguishable from untextured background in that its disparity values suggest a surface that is smooth and realistic (Figure 1, (bottom right)), not noisy and spiked like the one in Figure 1 (bottom left). Assuming we can identify all regions in an image that can be considered smooth physical surfaces, segmentation of the foreground is as simple as identifying all such surface regions that occur in front of the background. To find these surfaces, we use a modified connected components algorithm as follows:

1. consider DI to be a graph G where every vertex G_{ij} corresponds to a pixel DI_{ij}

2. for each vertex G_{ij} connect it to its four neighbors if and only if $\sum_{n \in N_{ij}} |G_{ij} - G_n| < t$ where N_{ij} is the 4-neighborhood of vertex G_{ij} and t is a user-defined threshold. Since neighboring disparity values in untextured regions differ by large amounts, there is a great deal of latitude in choosing this threshold.

3. connected components larger than a nominal size are accepted as surfaces and all other pixels are ignored. The size cutoff is also relatively easy to choose and is based on the assumption that the human figure will take up at least 10% of the image.

2.3 Segmentation Algorithm

Thus, our segmentation algorithm is:

1. using 20-30 images, model the background using a Gaussian $[\mu, \sigma]$ for each pixel

[1]In the rest of the paper, we refer to disparity images obtained using Digiclops; these are the depth images used in our experiments.

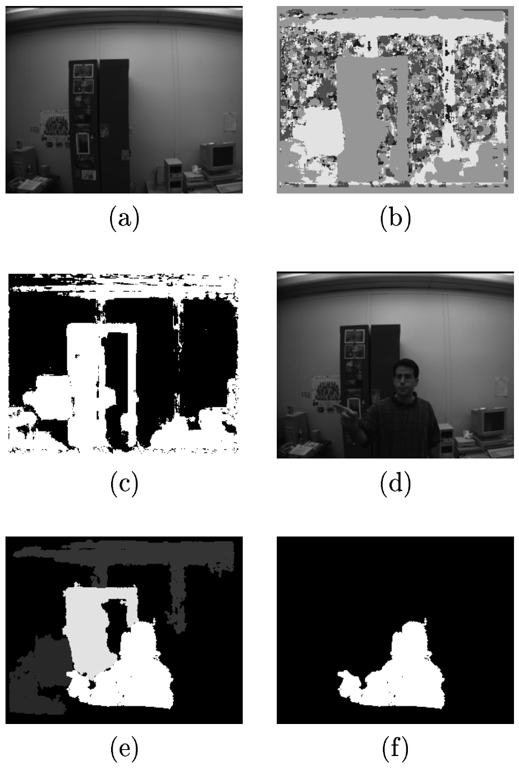


Figure 2: Illustration of segmentation: (a) One of the images in the input sequence of background images. (b) Disparity map output from Digiclops for this image. (c) Illustration of reliable (white) background pixels based on standard deviation of background model. (d) New, test image for foreground segmentation. (e) Results from surface validation. Surfaces shown (all regions except black) passed the validation requirement. (f) Final result of segmentation.

2. based on the values for standard deviation, determine unreliable background models by looking for pixels with $\sigma > \beta$ where β is a user-defined threshold (we used $\beta = 2$)

3. for each subsequent disparity image DI, calculate the areas considered to be physical surfaces

4. a surface pixel DI^s_{ij} is classified as foreground if

   (a) the background is reliable at $[i, j]$ and $DI^s_{ij} > \mu_{ij} + \sigma_{ij}$

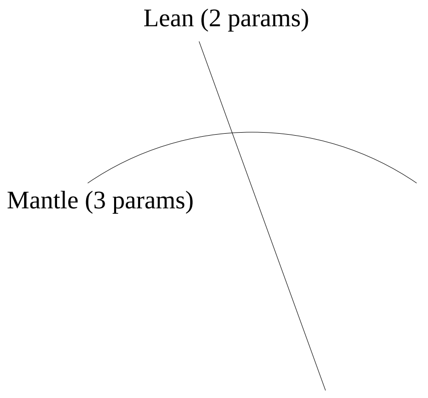
   (b) the background is unreliable at $[i, j]$

5. finally, to eliminate all remaining noise, we run a binary connected components algorithm and extract the largest component.

We've already looked at the first case in step 4, but the second deserves a bit of explanation. When we classify the background as unreliable, we are implicitly assuming that it is located in an untextured region. This is a safe assumption as it is generally only in those regions that the disparity value would fluctuate so much from frame to frame. As a result, the data in this region do not correspond to a physical surface and would not aggregate into a connected component large enough to classify as a surface. So, if we see a surface pixel where we expect to find an unreliable background pixel, we know that it must be part of the new foreground. Figure 2 illustrates these concepts.

3 Torso Model

Once we have an accurate segmentation, we look to fit a simple torso model to the foreground figure. The model is based on the assumption that the figure is upright or leaning slightly to one side. Given this, we notice that the occluding edges of the shoulders, heretofore referred to as the mantle, are strong cues that vary very little with respect to the motion of an upright figure. Thus our torso model consists only of five parameters, two for the straight line that captures the general lean of the figure and three for the quadratic that traces the outline of the mantle. Figure 3 shows the model and its application to a real image. We loosely interpret the intersection of the lean and the mantle as being the neck point. This will be useful later as we're looking to localize the head.



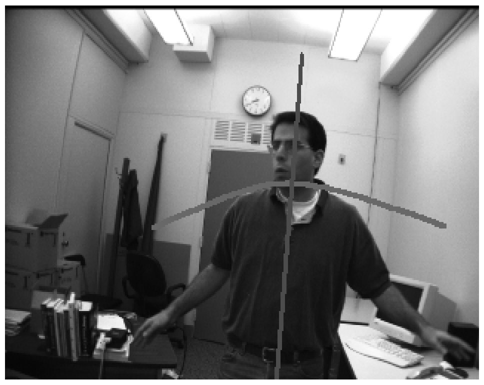


Figure 3: TOP: Illustration of simple torso model. BOTTOM: Application of torso model to image.

4 Model Acquisition

Because we have such a simple model, it is relatively easy to acquire. Given the segmented foreground figure as a binary image, we extract the figure's lean in the following way:

1. for each row of the binary image, calculate the median of the column values of the foreground pixels

2. using a Singular Value Decomposition line fit, find the best-fit line to the median values for each row. That line is the lean.

We use medians because they are less sensitive to outliers like waving arms. Since the width of a typical arm is far less than the width of a body, this works quite well as long as the lean angle is not too large. Figure 4 depicts an example of the lean acquisition.

Once we have the body lean, we can start acquisition of the mantle. As we mentioned before, the strongest cues are the occluding edges of the shoulders on either side of the head. To find these we employ

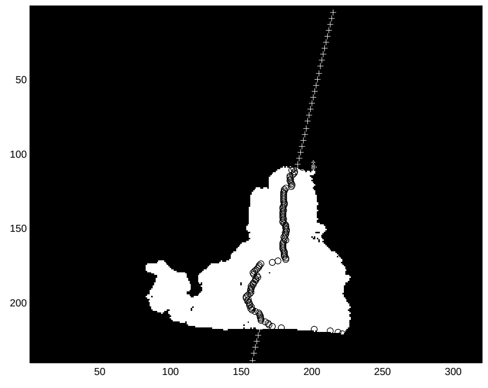


Figure 4: Using the medians of the rows of the foreground figure, we can extract the lean with an SVD line fit.

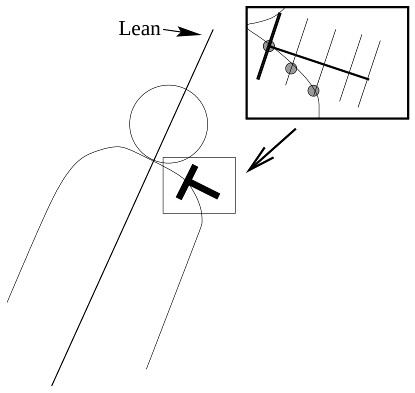


Figure 5: Example of a directed local edge detector. The detail of the figure shows the slices along which the image gradient is calculated. The gray dots represent occluding edge points found by the detector.

directed local edge detectors similar to those used in [13]. We orient these detectors perpendicular to the lean and look for edges by thresholding the image gradient along slices perpendicular to the detector's orientation (parallel to the lean). Figure 5 displays an example of such an edge detector.

To place these detectors in the best position, we use the depth data of the points along the lean to give us an estimate of how far the figure is from the camera. Based on this information, we can determine where, in image coordinates, to place the edge detectors. More specifically, we can decide where in 3D space we'd like to place the edge detectors. Then, using the focal lengths and baselines of the stereo cameras, we can determine where, in image coordinates, those would fall. For example, if the figure is close to the camera, we're going to look for edges along a much longer line than if it is further away. Figure 6 shows an illustration of these concepts. We use the assumption that the typical head is .2 meters wide and that the typical mantle is .4 meters across.

We place a series of these local edge detectors up and down the image perpendicular to the body lean and keep a running tally of how many potential edge points we find. After searching the length of the body lean line, we select the pair of trackers that yield the most edge points and, using least squares, fit a quadratic to those points. That quadratic is the mantle and represents the final three parameters of our model.

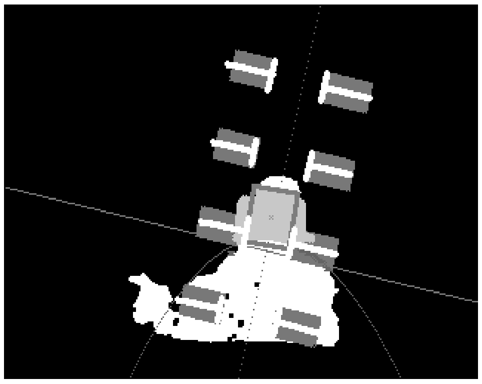


Figure 6: This image shows the lean, the mantle, and a few examples of the placement of the local edge detectors (gray rectangles). Also, the light grey area above the mantle line represents points classified as being part of the head. The 'x' represents the centroid (in image coordinates) of those head points.

5 Head Localization

Once we've acquired a model, we calculate the intersection of the mantle and the lean, which we interpret as the neck. We then look radially out from the neck at points in the foreground that are:

- 'above' the mantle
- within a reasonable distance (.2 meters) in world coordinates from the neck (remember we can do this because we are working with 3D data)

After identifying such points, we calculate their centroid and make the assumption that, regardless of tilt, this point will represent the center of the head. We can now determine the orientation of the head simply by calculating the angle made by the line containing the head's centroid and the neck point. We assume that the distance between those two points is half of the height of the head and can easily draw a box around it (Figure 6). Also, since we're using stereo, once we know the centroid of the head region we can easily figure out its position in 3D.

6 Results

Figures 7, 8 and 9 illustrate the results of our tracking scheme. Each sequence features cluttered backgrounds and rapid head movements that would be likely to confuse a tracker that relied on accurate predictions based on past motion. Plotted on each image is our acquired torso model as well as the orientation of the head. Figure 8 especially shows the tracker's ability to work even in the presence of waving arms and

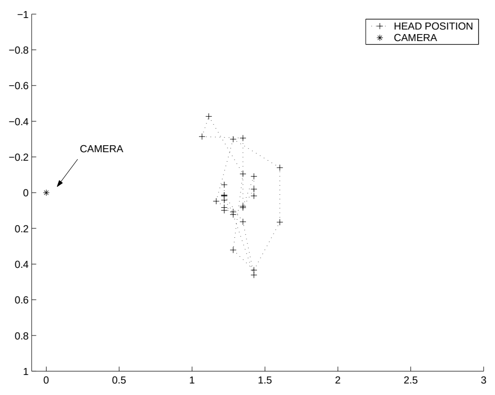


Figure 10: Bird's eye view of head movement through room in sequence from Figure 7. Both axes are in meters.

image clutter. It also shows one of the failure modes of the system. When the assumption that the figure more or less faces the camera is violated, the shoulder cues are not always strong enough to lead us to the correct configuration of the model. Fortunately, since our next step is entirely independent of the previous one, we are not confused for long and reacquire the figure soon after. Similarly, in Figure 9 the tracker gets confused when the figure's arms occlude the head and shoulders. As soon as the occlusion ends, however, the tracker immediately reaquires its target. Figure 10 shows one of the important side effects from using stereo. Since we are using stereo and we know the cameras' intrinsic and extrinsic parameters, once we find where the head is located in image coordinates, we can easily turn that into a 3D point. As a result, we can track the movement of the head throughout a room in 3D.

6.1 Performance

This system is run using a resolution of 320x240 pixels and the processing time per frame is approximately one second on a dual Pentium II 350 MHz.

7 Conclusion

What we have shown is a new approach to head tracking taking advantage of the segmentation accuracy real-time stereo affords. We've created a simple torso model that is quick to acquire and does not require accurate predictions between frames to work. As a result, we can ignore the common assumption of small interframe motions as well as the problems generated by occlusions. We use this system to track heads in 3D throughout a room.

7.1 Future Work

We hope to use this algorithm in several ways. For one, this is an important complement to a smart room where knowing the 3D position of a user's head is a way to steer a microphone array to listen in that direction. We would also like to use it as a first step in the bootstrapping of a more complicated articulated motion tracking system that can perform human gesture recognition. In addition, work is underway to use this system to provide real-time 3D head coordinates as an input to a face recognition algorithm similar to the one in [2].

As for extensions of the algorithm itself, a relatively easy one would be to track multiple heads in an image. This is a simple matter of identifying all of the blobs in the foreground and acquiring a torso model for each. Another extension under consideration is to add a very simple prediction step that would reduce the computation time to acquire a model, but not sacrifice the robustness of our 'one image at a time' system.

Acknowledgments

The authors would like the thank Michael Benharrosh and Vince Stanford for many fruitful discussions.

References

[1] T. Darrell, B. Blumberg, S. Daniel, B. Rhodes, P. Maes, and A. Pentland, "Alive: Dreams and Illusions," *ACM SIGGraph, Computer Graphics Visual Proceedings*, July, 1995.

[2] T. Darrell, B. Moghaddam, and A. Pentland, "Active Face Tracking and Pose Estimation in an Interactive Room," *IEEE Conference on Computer Vision and Pattern Recognition*, June, 1996.

[3] S. Birchfield, "An Elliptical Head Tracker," *31st Asilomar Conference on Signals, Systems, and Computers*, pp. 1710-1714, November, 1997

[4] S. Birchfield, "Elliptical Head Tracking Using Intensity Gradients and Color Histograms," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 232-237, June, 1998.

[5] M. La Cascia, J. Isidoro, and S. Sclaroff, "Head Tracking via Robust Registration in Texture Map Images," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 508-514, June, 1998.

[6] S. Basu, I. Essa, and A. Pentland, "Motion Regularization for Model-based Head Tracking," *Proceedings of the International Conference on Pattern Recognition*, August, 1996.

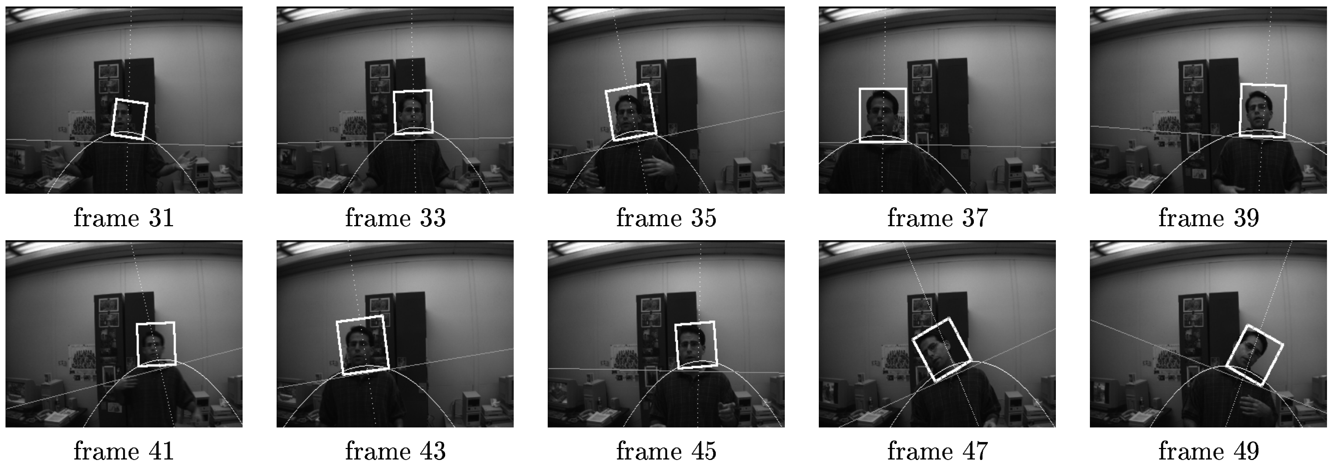


Figure 7: Results from a sequence of rapid head movements. The images were acquired at about 2 Hz so this sequence lasts about 10 seconds.

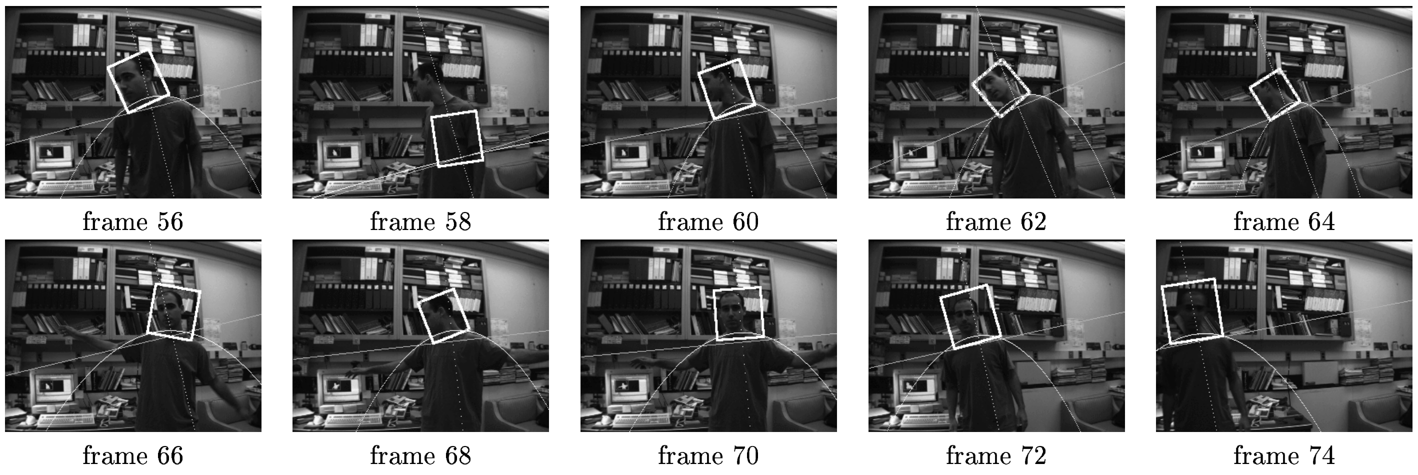


Figure 8: Results from another rapid sequence of head and arm movements with a cluttered background. Notice the failure of the tracker (frame 58) when the figure is turned too much to the side reducing the strength of the shoulder cues. The timing is the same as in Figure 7

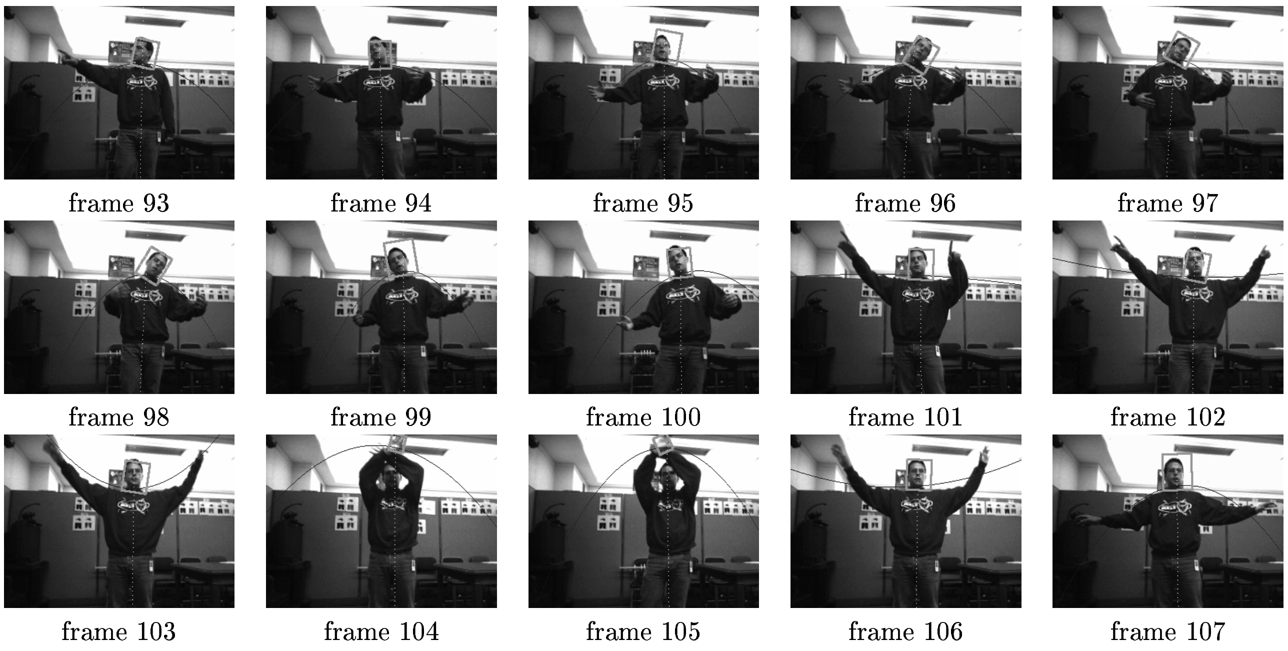


Figure 9: Results from a third sequence of rapid head and arm movements background. Notice the failure of the tracker (frames 104, 105) when the arms occlude the figure. When the occlusion disappears, the tracker immediately reacquires the head. The timing is the same as in Figure 7

[7] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-Time Tracking of the Human Body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, pp. 780-785, July, 1997.

[8] A. Azarbayejani, C. Wren, and A. Pentland, "Real-time 3-D Tracking of the Human Body," *Proceedings of IMAGE'COM 96*, May, 1996.

[9] S. Niyogi and W. Freeman, "Example-Based Head Tracking," *IEEE 2nd Intl. Conf. on Automatic Face and Gesture Recognition*, October, 1996.

[10] K. Konolige, "Small Vision Systems: Hardware and Implementation," *Eighth Interational Symposium on Robotics Research*, October, 1997.

[11] T. Darrell, G. Gordon, J. Woodfill, and M. Harville, "Integrated person tracking using stereo, color, and pattern detection," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 601-609, June, 1998.

[12] C. Eveland, K. Konolidge, and R.C. Bolles, "Background Modeling for Segmentation of Video-Rate Stereo Sequences," *IEEE Conference on Computer Vision and Pattern Recognition*, June, 1998.

[13] J. Rehg and T. Kanade, "Visual Tracking of High DOF Articulated Structures: an Application to Human Hand Tracking," *Third European Conference on Computer Vision*, pp. 35-46, May, 1994.

[14] N. Jojic, M. Turk, and T. Huang, "Tracking Self-Occluded Articulated Objects in Dense Disparity Maps," *International Conference on Computer Vision*, pp. 123-130, September, 1999.

[15] M. Turk, "Visual Interaction With Lifelike Characters," *IEEE 2nd Intl. Conf. on Automatic Face and Gesture Recognition*, October, 1996.